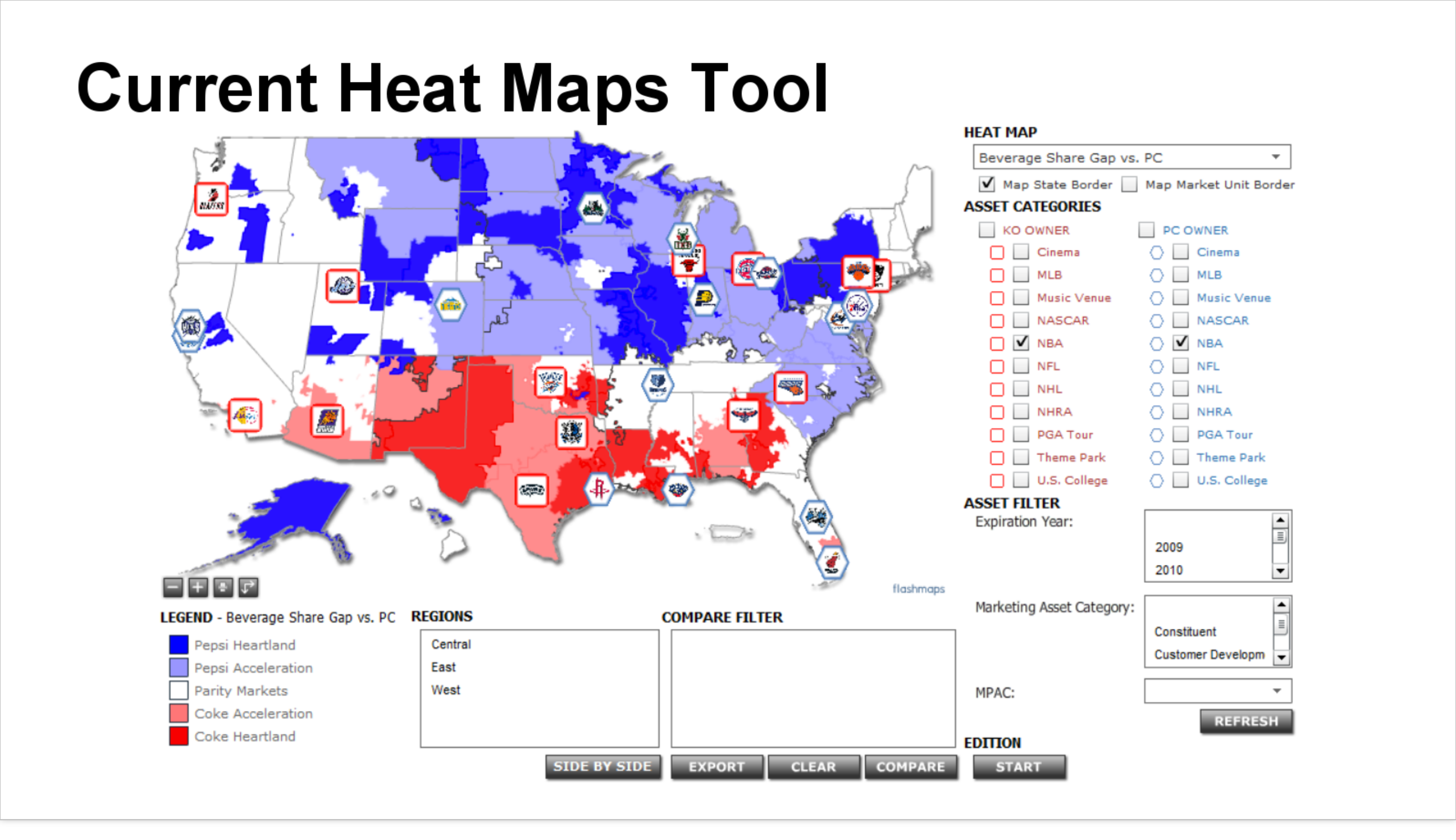


# Current Heat Maps Tool

**HEAT MAP**

Beverage Share Gap vs. PC

☑ Map State Border ☐ Map Market Unit Border

**ASSET CATEGORIES**

**ASSET FILTER**

Expiration Year: 2009 2010

Marketing Asset Category: Constituent Customer Developm

MPAC:

**LEGEND** - Beverage Share Gap vs. PC

- Pepsi Heartland
- Pepsi Acceleration
- Parity Markets
- Coke Acceleration
- Coke Heartland

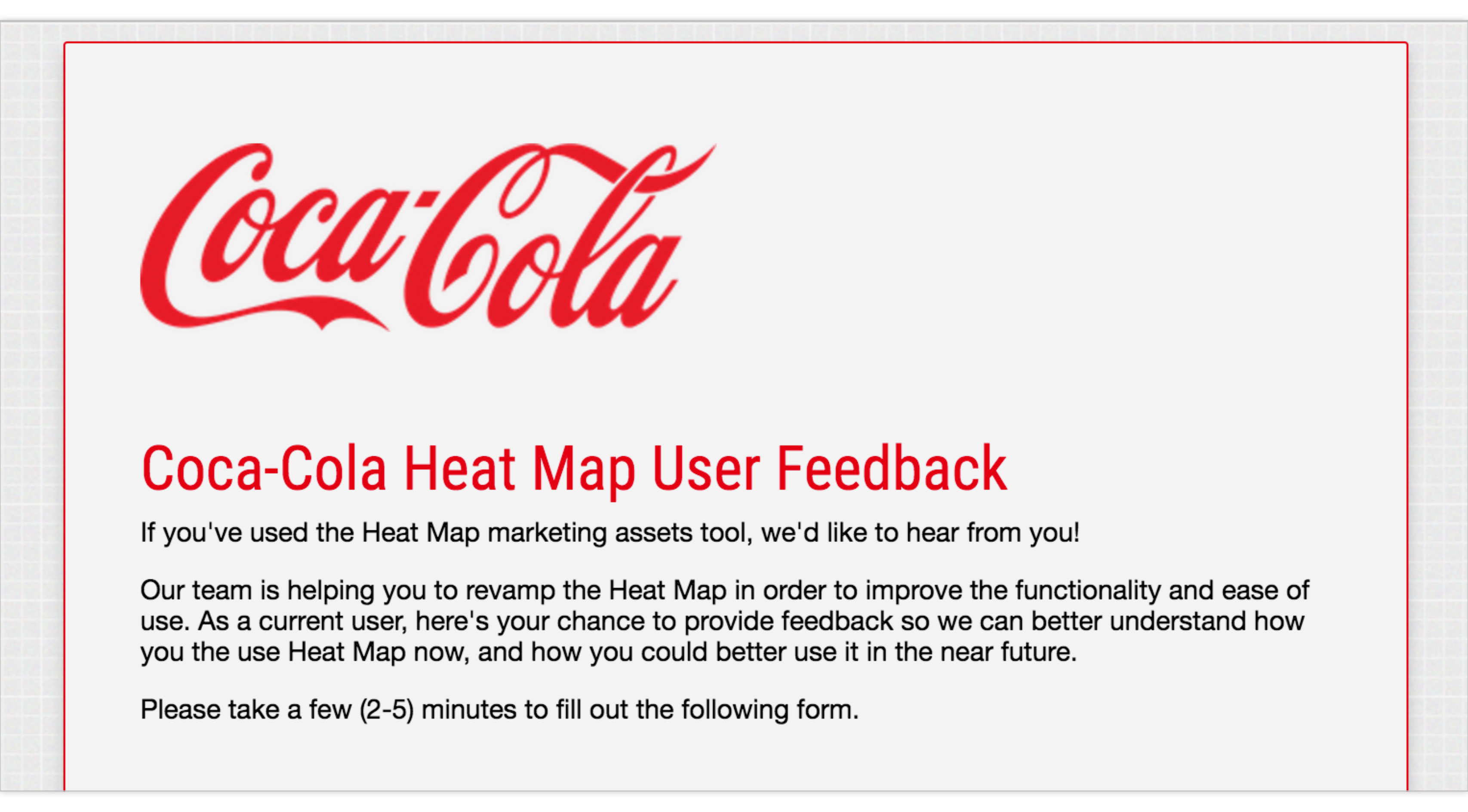
**REGIONS**

Central
East
West

**COMPARE FILTER**

**EDITION**

The starting point — an outdated Flash application. Our object was first, to understand the tool's purpose and valuable features. Then second, to redesign and rebuild the tool while making it more effective, efficient, and pleasant to use.



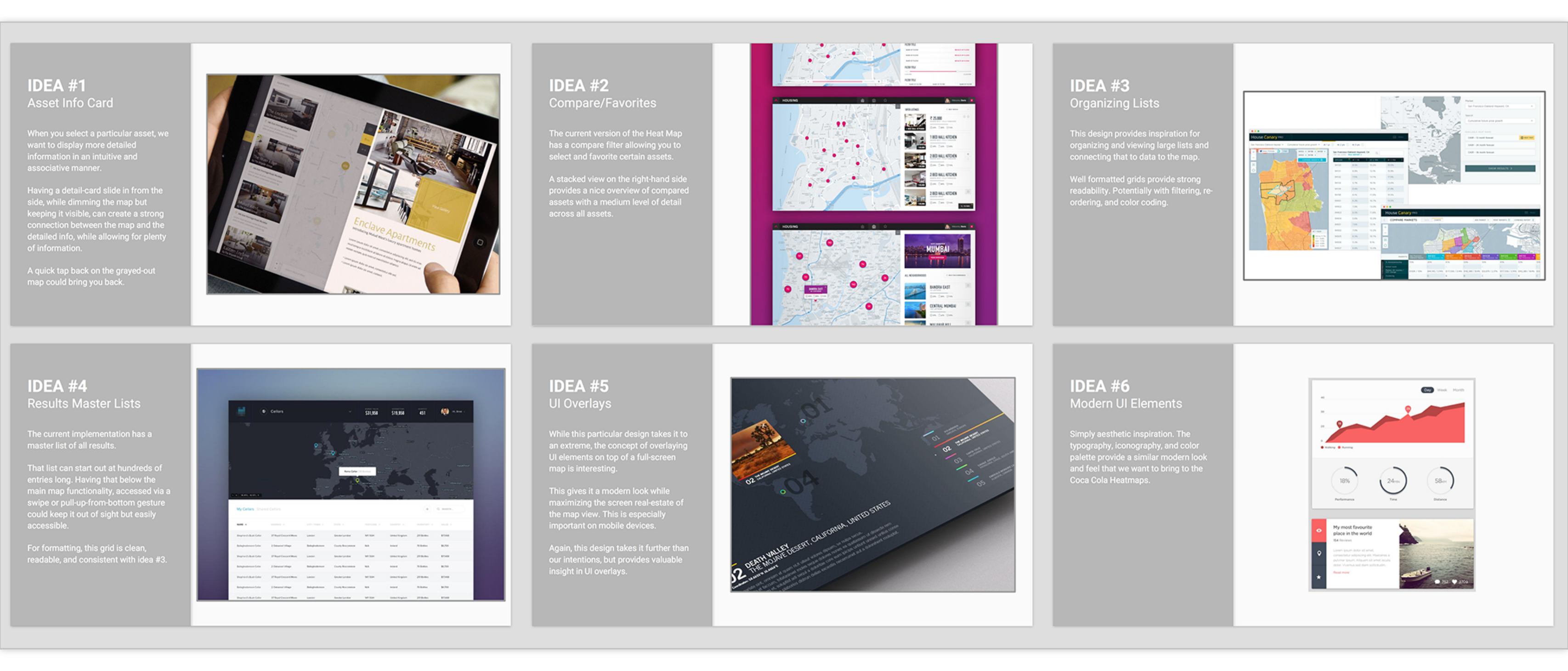
## Coca-Cola Heat Map User Feedback

If you've used the Heat Map marketing assets tool, we'd like to hear from you!

Our team is helping you to revamp the Heat Map in order to improve the functionality and ease of use. As a current user, here's your chance to provide feedback so we can better understand how you the use Heat Map now, and how you could better use it in the near future.

Please take a few (2-5) minutes to fill out the following form.

We begin with UX research. By gathering information from users, we can understand more precisely how the tool currently works, and what users expect the tool to do. We inteviewed key stakeholders for quality feedback, and provided this form for quantitative feedback.



**IDEA #1**
Asset Info Card

When you select a particular asset, we want to display more detailed information in an intuitive and associative manner.

Having a detail-card slide in from the side, while dimming the map but keeping it visible, can create a strong connection between the map and the detailed info, while allowing for plenty of information.

A quick tap back on the grayed-out map could bring you back.

**IDEA #2**
Compare/Favorites

The current version of the Heat Map has a compare filter allowing you to select and favorite certain assets.

A stacked view on the right-hand side provides a nice overview of compared assets with a medium level of detail across all assets.

**IDEA #3**
Organizing Lists

This design provides inspiration for organizing and viewing large lists and connecting that to data to the map.

Well formatted grids provide strong readability. Potentially with filtering, re-ordering, and color coding.

**IDEA #4**
Results Master Lists

The current implementation has a master list of all results.

That list can start out at hundreds of entries long. Having that below the main map functionality, accessed via a swipe or pull-up-from-bottom gesture could keep it out of sight but easily accessible.

For formatting, this grid is clean, readable, and consistent with idea #3.

**IDEA #5**
UI Overlays

While this particular design takes it to an extreme, the concept of overlaying UI elements on top of a full-screen map is interesting.

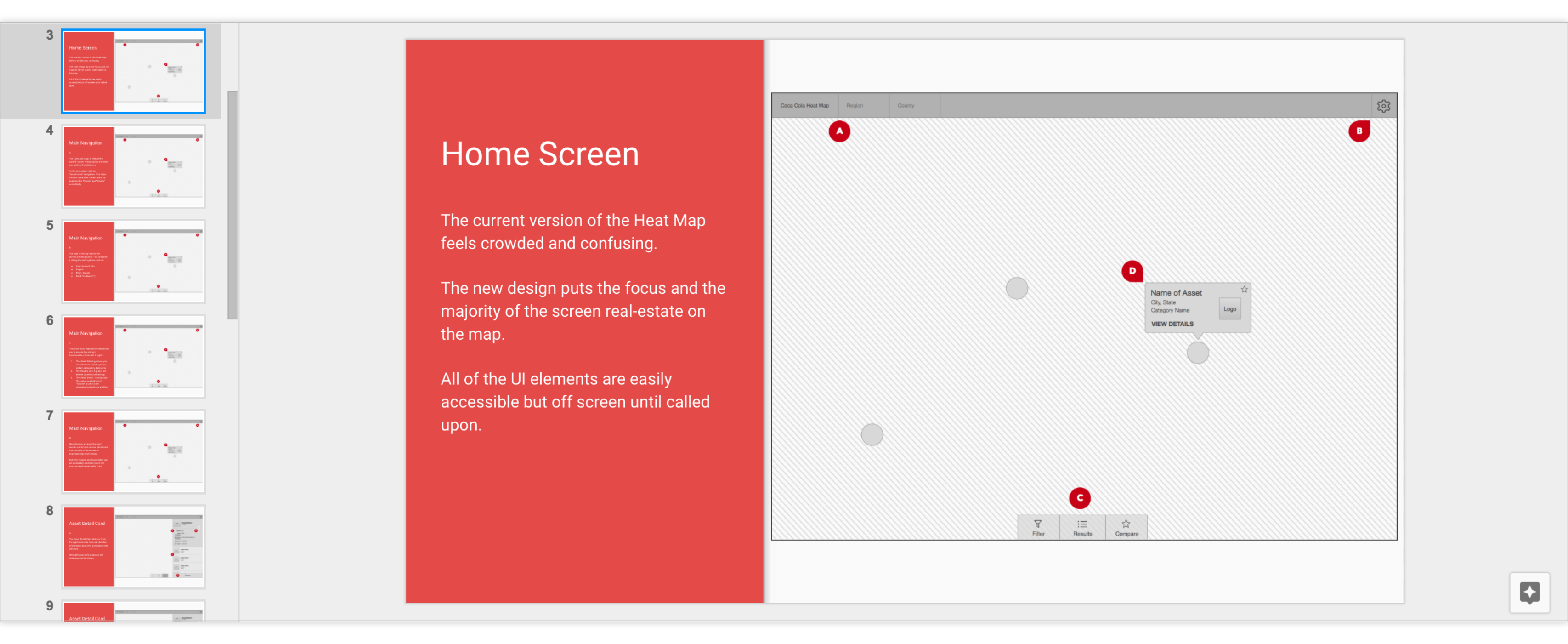
This gives it a modern look while maximizing the screen real-estate of the map view. This is especially important on mobile devices.

Again, this design takes it further than our intentions, but provides valuable insight in UI overlays.

**IDEA #6**
Modern UI Elements

Simply aesthetic inspiration. The typography, iconography, and color palette provide a similar modern look and feel that we want to bring to the Coca Cola Heatmaps.

After diagnosing the key features, we searched for design proof of concepts. We presented the new design solutions and explained the reasoning for our approach. Design with intent — one should always be able to explain their design reasoning.



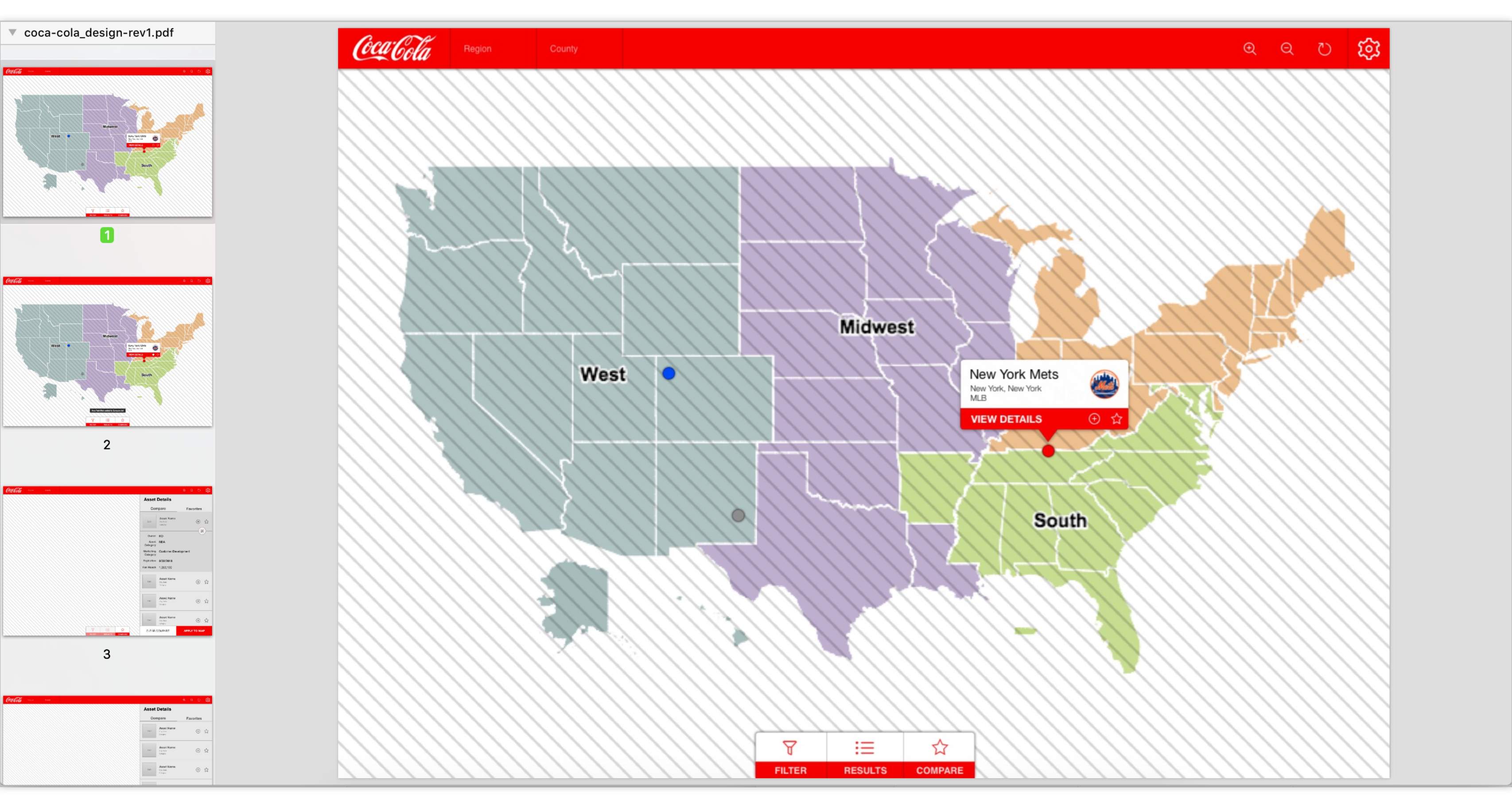
**Home Screen**

The current version of the Heat Map feels crowded and confusing.

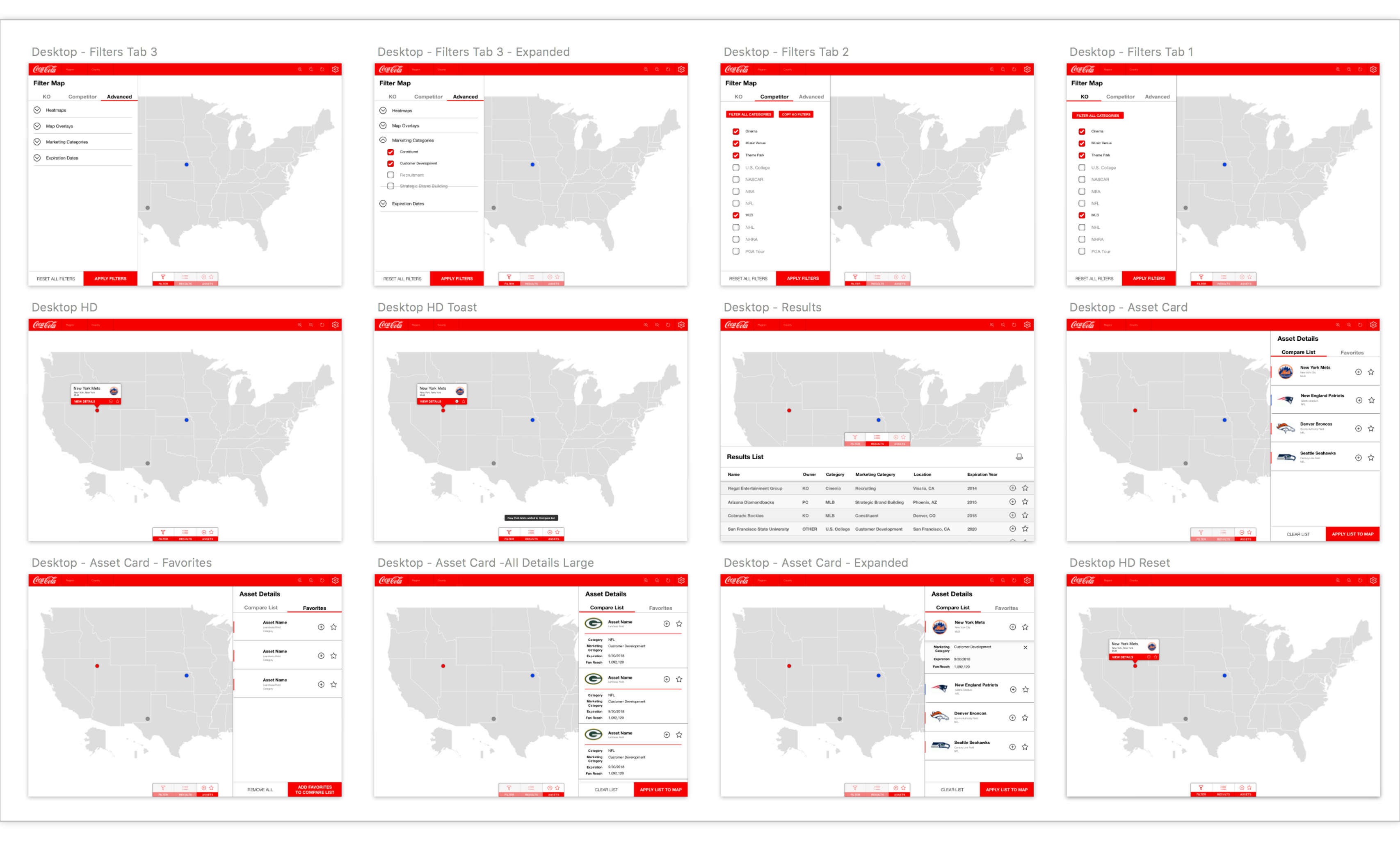
The new design puts the focus and the majority of the screen real-estate on the map.

All of the UI elements are easily accessible but off screen until called upon.

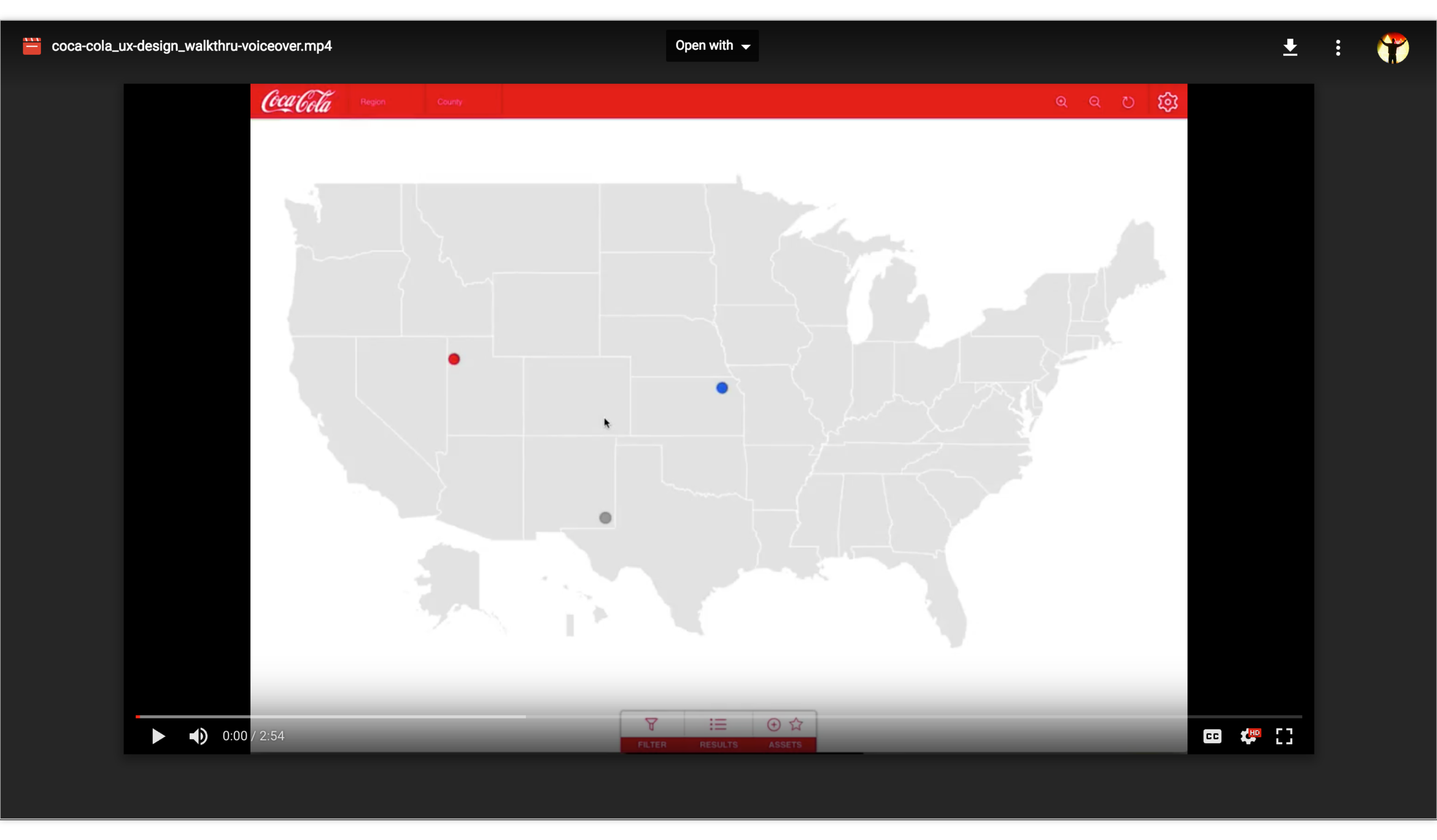
With general ideas agreed upon, we began with rough UI wireframes. In this early stage we were laying the foundation for the rest of the design process, so we again thought it imperative we shared our drafts with the clients and elaborated our reasoning.



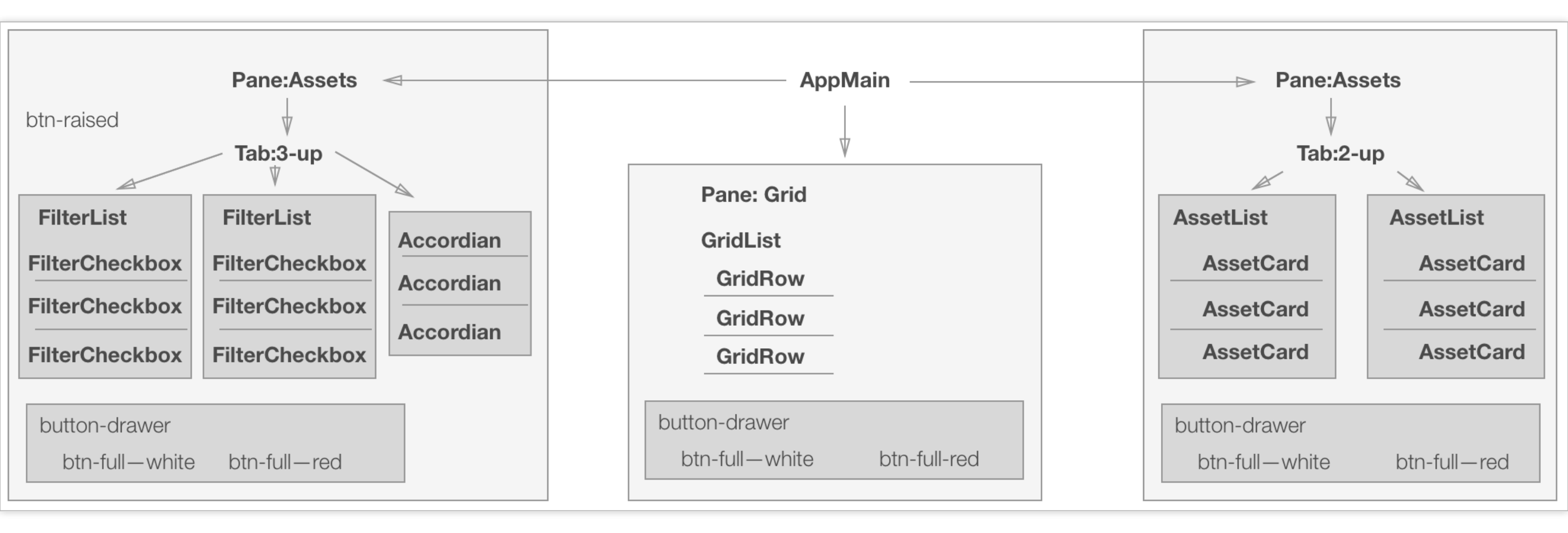
Wireframes begin to take form by solidifying features and incorporating the client's Branding Documents.



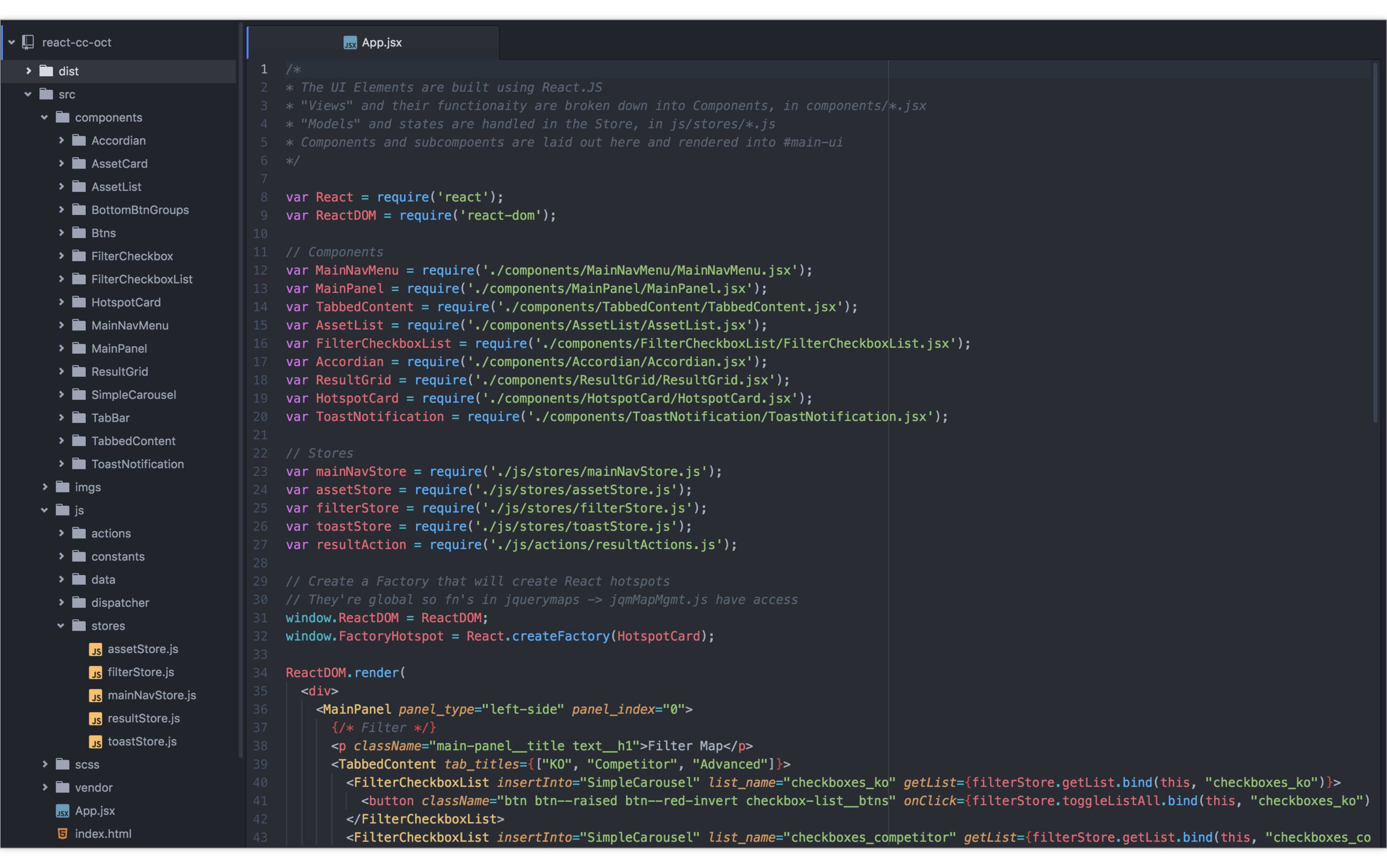
A few rounds of feedback revisions between designers and the client's key stakeholders, and we've got finalized designs for the application's user interface.



Static UI assets can be difficult to visualize. To tackle that problem, I whipped up an interactive and animated prototype. By screen capturing a walk-through, we efficiently provided the client a better understanding of how the new UI would work — before deep diving into code.



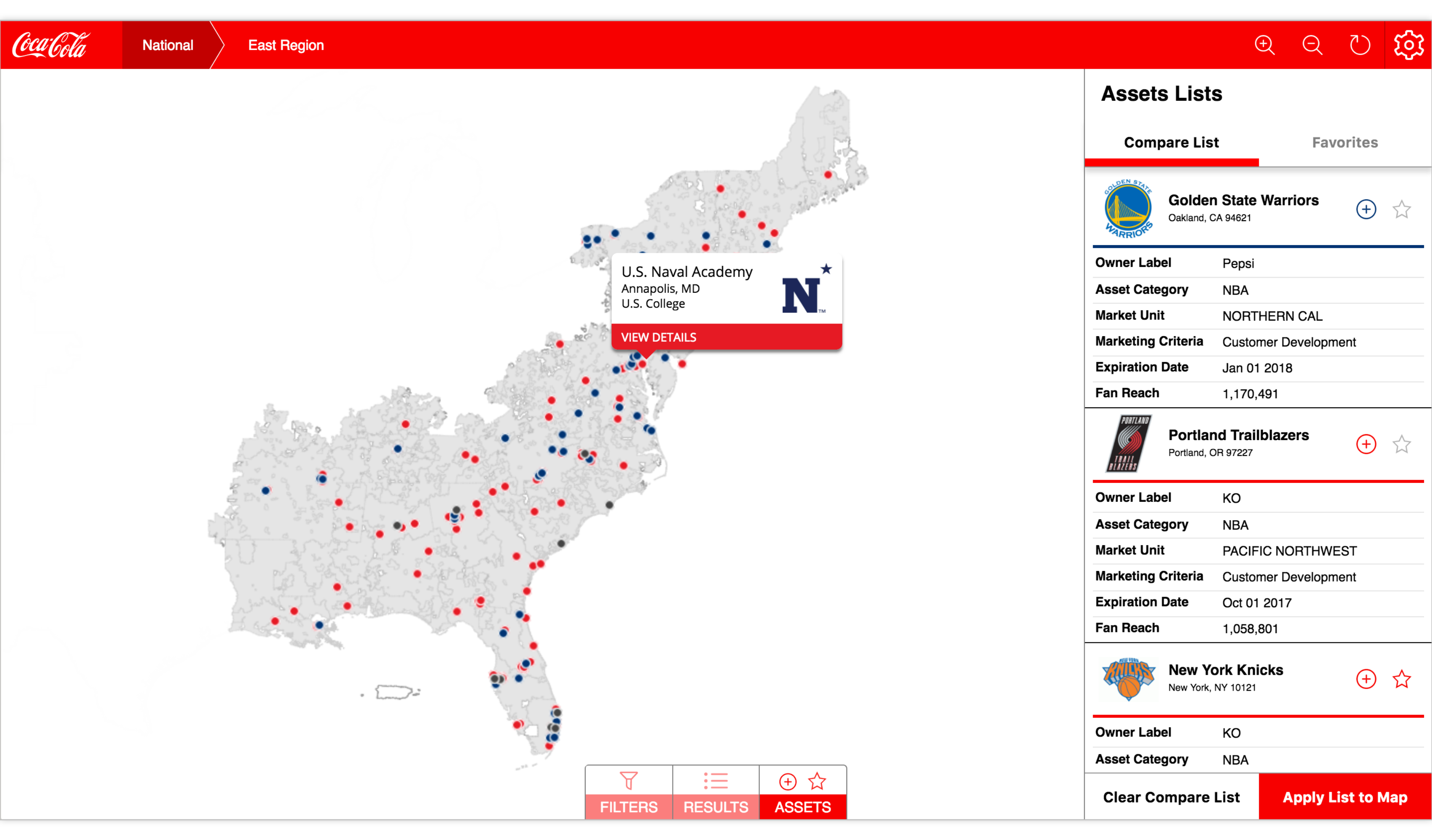
First step before writing a line of code? Plan. We mapped out the application architecture, providing a clear understanding of the components to be built and where they could be reused.



App.jsx

```
/*
* The UI Elements are built using React.JS
* "Views" and their functionaity are broken down into Components, in components/*.jsx
* "Models" and states are handled in the Store, in js/stores/*.js
* Components and subcompoents are laid out here and rendered into #main-ui
*/

var React = require('react');
var ReactDOM = require('react-dom');

// Components
var MainNavMenu = require('./components/MainNavMenu/MainNavMenu.jsx');
var MainPanel = require('./components/MainPanel/MainPanel.jsx');
var TabbedContent = require('./components/TabbedContent/TabbedContent.jsx');
var AssetList = require('./components/AssetList/AssetList.jsx');
var FilterCheckboxList = require('./components/FilterCheckboxList/FilterCheckboxList.jsx');
var Accordian = require('./components/Accordian/Accordian.jsx');
var ResultGrid = require('./components/ResultGrid/ResultGrid.jsx');
var HotspotCard = require('./components/HotspotCard/HotspotCard.jsx');
var ToastNotification = require('./components/ToastNotification/ToastNotification.jsx');

// Stores
var mainNavStore = require('./js/stores/mainNavStore.js');
var assetStore = require('./js/stores/assetStore.js');
var filterStore = require('./js/stores/filterStore.js');
var toastStore = require('./js/stores/toastStore.js');
var resultAction = require('./js/actions/resultActions.js');

// Create a Factory that will create React hotspots
// They're global so fn's in jquerymaps -> jqmMapMgmt.js have access
window.ReactDOM = ReactDOM;
window.FactoryHotspot = React.createFactory(HotspotCard);

ReactDOM.render(
  <div>
    <MainPanel panel_type="left-side" panel_index="0">
      {/* Filter */}
      <p className="main-panel__title text__h1">Filter Map</p>
      <TabbedContent tab_titles={["KO", "Competitor", "Advanced"]}>
        <FilterCheckboxList insertInto="SimpleCarousel" list_name="checkboxes_ko" getList={filterStore.getList.bind(this, "checkboxes_ko")}>
          <button className="btn btn--raised btn--red-invert checkbox-list__btns" onClick={filterStore.toggleListAll.bind(this, "checkboxes_ko")
        </FilterCheckboxList>
        <FilterCheckboxList insertInto="SimpleCarousel" list_name="checkboxes_competitor" getList={filterStore.getList.bind(this, "checkboxes_co
```

The tool was a responsive, data-driven, UI-heavy web application. We decided the best framework was ReactJS for front-end UI components, and Flux for managing application state and connecting to Database APIs. We also had the challenge of integrating a 3rd-party map API.



U.S. Naval Academy
Annapolis, MD
U.S. College
VIEW DETAILS

**Assets Lists**

Compare List Favorites

**Golden State Warriors**
Oakland, CA 94621

| | |
|---|---|
| Owner Label | Pepsi |
| Asset Category | NBA |
| Market Unit | NORTHERN CAL |
| Marketing Criteria | Customer Development |
| Expiration Date | Jan 01 2018 |
| Fan Reach | 1,170,491 |

**Portland Trailblazers**
Portland, OR 97227

| | |
|---|---|
| Owner Label | KO |
| Asset Category | NBA |
| Market Unit | PACIFIC NORTHWEST |
| Marketing Criteria | Customer Development |
| Expiration Date | Oct 01 2017 |
| Fan Reach | 1,058,801 |

**New York Knicks**
New York, NY 10121

| | |
|---|---|
| Owner Label | KO |
| Asset Category | NBA |

Clear Compare List Apply List to Map

The final result — a clean, focused, and useful update to the Heat Map tool. The client's feedback was mission successful. All of the core features were there, but now in a more usable design and on a modern technology stack.